FIG.1

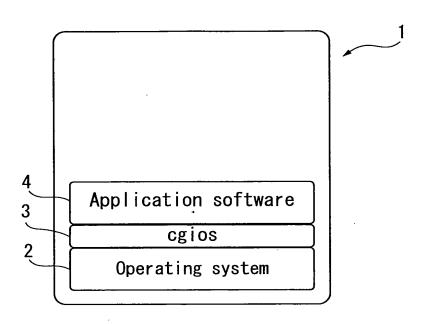
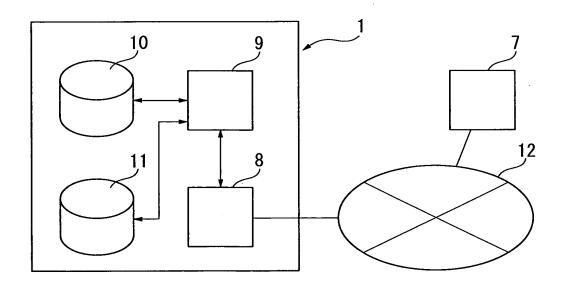


FIG.2



Invoke processing command for creating second HTML document Invoke processing command for Invoke processing command for creating O-th HTML document creating first HTML document (DB registration completion) Register input data stored bage) from object program ★Second processing step ★O-th processing step ★First processing step (order initiation page) in data recording part in database. Stream program from object program. from object program. (data entry form) (processing end on setting item of input data from reply information and compare. Generate and encrypt new session value and random number value, append them to first HTML document and send it to client. are identical & setting items are identical ightarrow input input-data Detect session value, random number value and information · When session values are identical & random number values into data recording part and execute second processing step. Invoke processing command in first processing step from stream program to create first HTML document. Record session value, random number and processing step information etc. on recording portion. Record session value, random number and processing step Execute first processing step when session values are (Release control portion from service processing) (Release control portion from service processing) (Release control portion to service processing) (Release control portion to service processing) identical & random number values are different. Similar processing as described above. information etc. on recording portion cgios (Start of service processing) (End of service processing) Similar hereinafter Network Jusilo

```
ustream input_order($0) { # order entry stream
  top:
    $o = new <APP>::OysterOrder($S);
    $o->default;
  first:
    page(input_order.page) {
        V={SLOT} = { slot => $0-> input };
    if ($V->{action} eq "next") { goto second; }
    goto first;
  second:
    if (\$o->\{ERROR\} = \$o->check) \{goto first;\}
    page(print_order_conf.page) {
        V={SLOT} = { slot => $0->print };
    }
    if ($V->{action} eq "back") { goto first; }
    elsif ($V->{action} eq "next") { goto third; }
    goto second;
  third:
    0-> {orid} = 0-> nextorid;
    $$${DA}->saveObject($0);
  fourth:
    page(print_order_done.page) {
        V=\{SLOT\} = \{ slot => $o->print \};
    if ($V->{action} eq "next") { goto first; }
    goto fourth;
}
```

```
# Object that represents order of oyster dish. Detail of member variable is
<MEMBER>.
package OysterOrder;
# Object that represents order of oyster dish
#BEGIN_DECLARE
                                         # Order ID
$data = "orid int,
                                           # Oyster's brand code
        brand int,
                                          # Recipe code
        recipe int,
                                          # Seasoning code
        trimming int,
        comment text,
                                             # Comment regarding recipe
                                                                                      рl
        primary key(orid)";
$seqs = "seq_orid:orid:1,";
$defo = "brand => OysterType, recipe => RecipeType, trimming => TrimmingType";
#END_DECLARE
use strict:
sub new { # Constructor of oyster dish ordering object
    my(type, S, \%hash) = @_{j}
                                                                                      p2
    my %h = ( "S" => $S, "SELF' => "OysterOrder", %hash);
    return bless \pmu \mathbb{\text{Nh}}, "<\text{APP}>::OysterOrder";
}
sub nextorid { # Next ORID
    my($V) = @_: my $S = $V->{S};
    my($orid) = $$S{DA}->getrow("select nextval("seq_orid")");
    $orid;
}
sub default { # Set default value
    my($V) = @_: my $S = $V->{S};
    V->\{brand\} = 1; \# Rock oyster
    $V->{recipe} = 1; # Eaten-raw
    $V->{trimming} = 2; # Lemon only
    $V->{comment} = "";
    delete $V->{orid};
}
```

```
sub input { # Order entry form
  my($V) = @ : my $S = $V->{S};
  print_webform( # Order entry form
'
<ifdef ERROR><span class=error><!ERROR></span></ifdef>
Brand of oyster
  Recipe of oyster 
 <input type=select name=recipe value=$recipe choice=[RecipeType]>
Trimmed seasoning
 <input type=TrimmingType name=trimming value=$trimming>
');
}
sub check {
  my($V) = @_: my $S = $V->{S};
  my $count = 0;
  my $value = $V->{trimming};
  while ($value>0) { $count += $value & 1; $value >>= 1; }
                                                        р6
  my \$error = "";
  if (\text{sount} >= 3) {
     $error = "You can only attach up to two kinds of seasoning at once.";
  $error;
}
sub print { # Fair copy of order in HTML format
  my($V) = @_: my $S = $V -> {S};
  print_webtext( # Fair copy of order in HTML format
'
Brand of oyster <%brand>
                                                         р7
 Recipe of oyster<%recipe>
Trimmed seasoning%trimming>
Comment<%comment>
');
```

```
(a) < Page for inputting order:/main/input_order.page >
       <h3> Please select brand of oyster and recipe</h3>
       <#slot>
       <br>
   <input type=submit action="next" label=" NEXT ">
 (b) < Page for confirming order:/main/print_order_conf.page >
       <h3> Please confirm your order.</h3>
       <#slot>
       <br>
       <input type=submit action="next" label=" ORDER EXECUTE'>
<input type=submit action="back" label="BACK">
 (c) < Page for notifying that order is completed:/main/print_order_done.page >
       <h3> The following order has been accepted.</h3>
       <#slot>
       <br>
<input type=submit action='next' label="BACK TO BEGINNING">
```

FIG.8

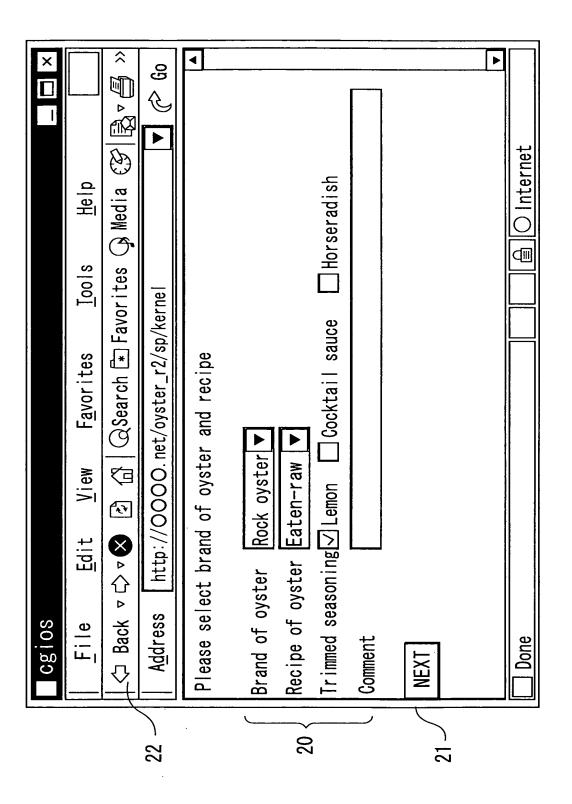


FIG.9

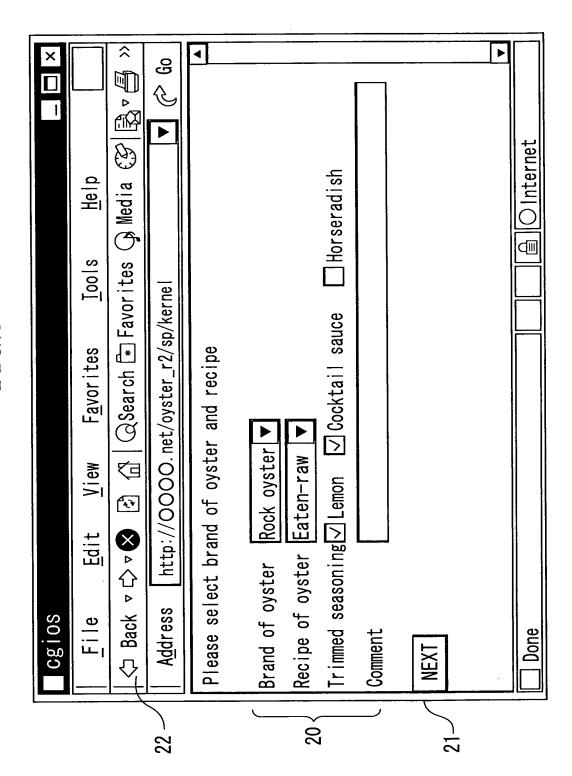
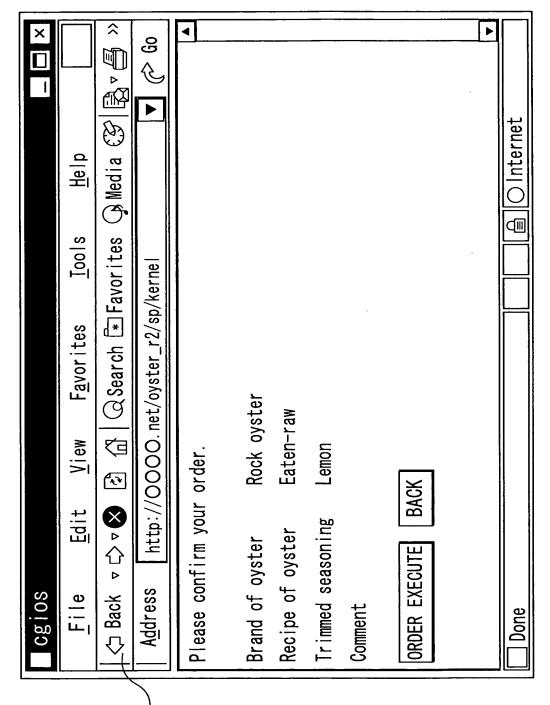


FIG.10



23

FIG.11

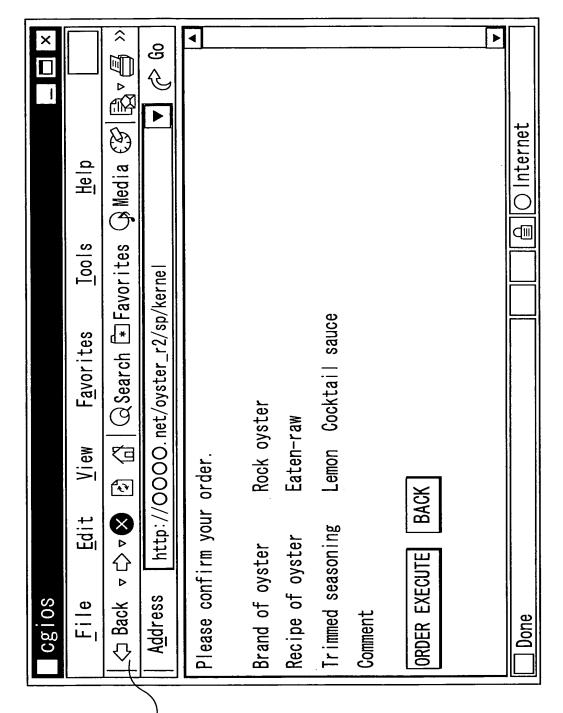


FIG.12

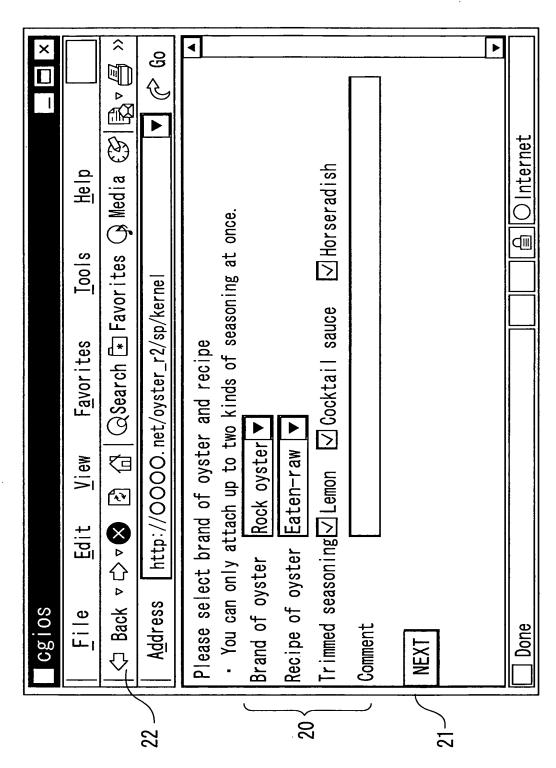


FIG.13

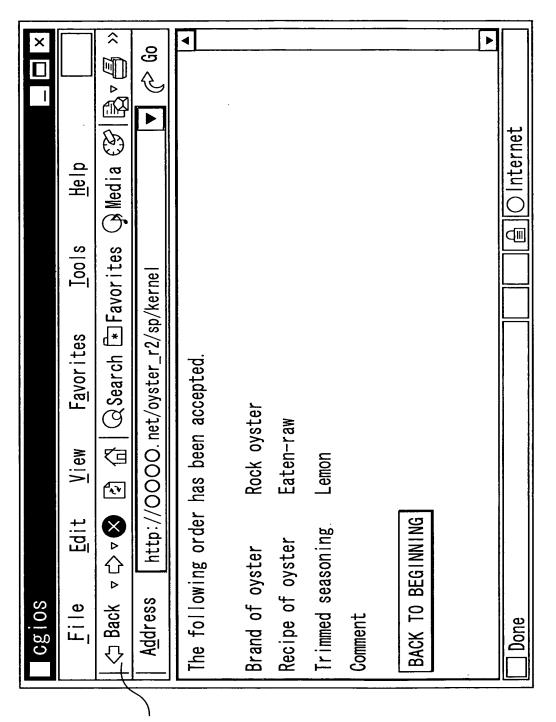


FIG.14

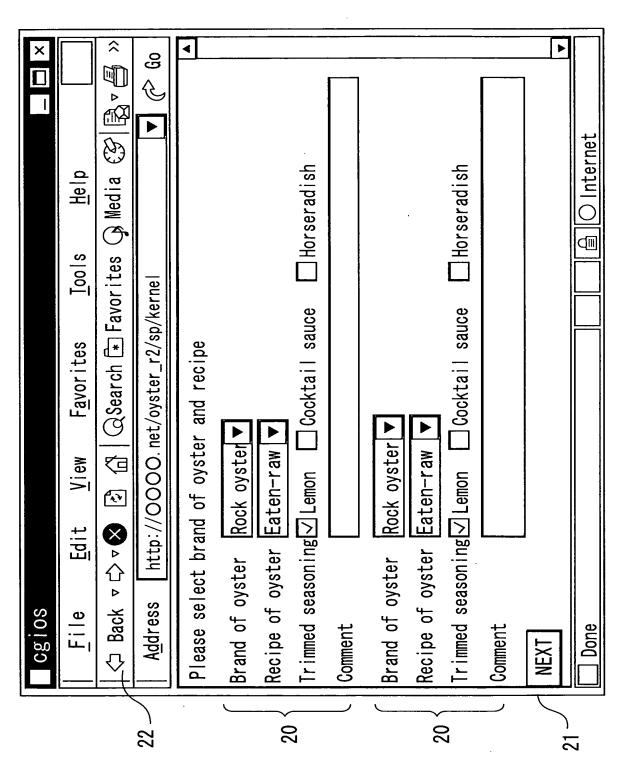
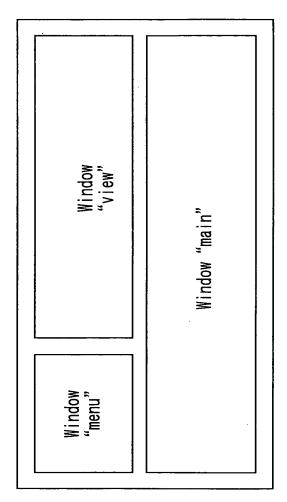


FIG.15



Frame "/main/top.page"

```
(a) <Activation stream:/init.stream>
        ustream init() { # Activation stream
         loop:
          p_fork("Window stream","/main/init.stream","this");
          p_wait();
          goto loop;
}
 (b) < Window activation stream:/main/init.stream>
        ustream init() { # Window stream
          loop:
            p_setPage("top.page");
            p_fork("menu","menu.stream","menu");
            p_wait();
            goto loop;
}
 (c) <Menu stream:/main/menu.stream>
        ustream menu() { # Menu display stream
          first:
            page(menu.page) {
            if ($V->{action} eq 'input_order') {
                p_fork("INPUT ORDER", 'input_order.stream", ''^view');
            } elsif ($V->{action} eq '1ist') {
                p_fork("DISPLAY ORDERING LIST", "list.stream", "^main");
            goto first;
}
```

```
(a) <Top frame (new addition):/main/top.page>
    cellpadding=0 style='table-layout:fixed'>
     >
     <div style="overflow:auto"><#menu></div>
     <div style="overflow:auto"><#view></div>
     <div style="overflow:auto"><#main></div>
     (b) <Menu page (new addition):/main/menu.page>
    <h3> Main menu </h3>
    <table
         class=inner
                 bgcolor=#fffff
                          border-color=#000000
                                         border=2
cellspacing=0 cellpadding=10>
    <a href="action:input_order">INPUT ORDER</a>
    <a href="action:list"> DISPLAY ORDERING LIST</a>
```

FIG.18

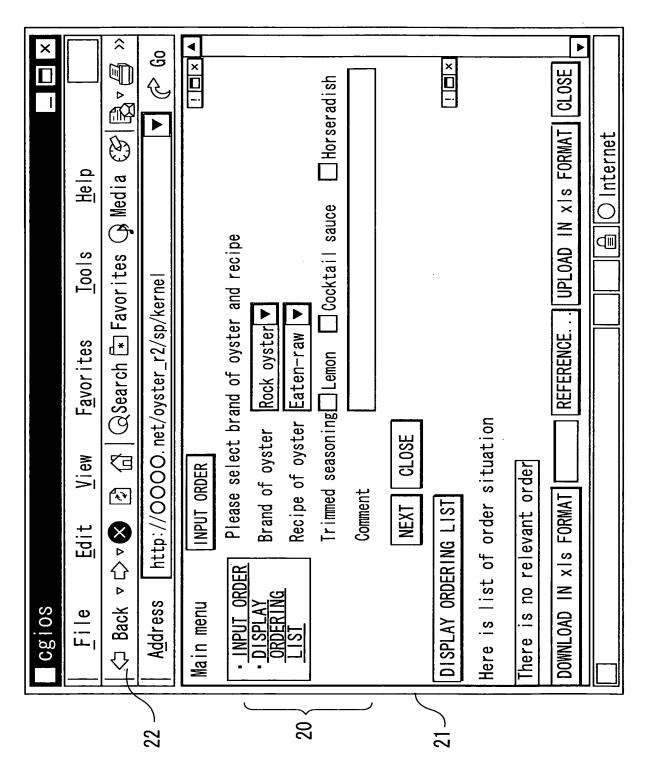
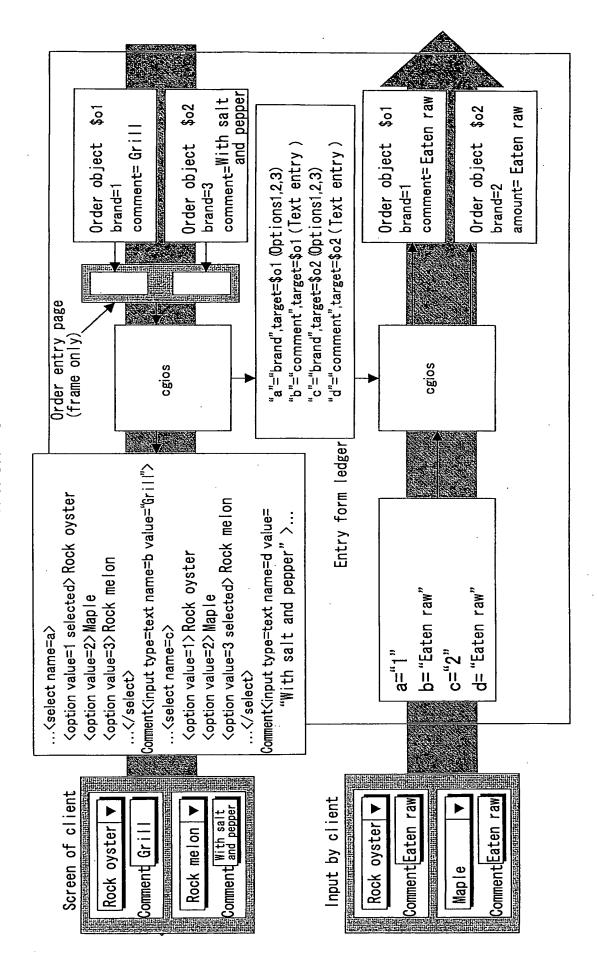


FIG.19





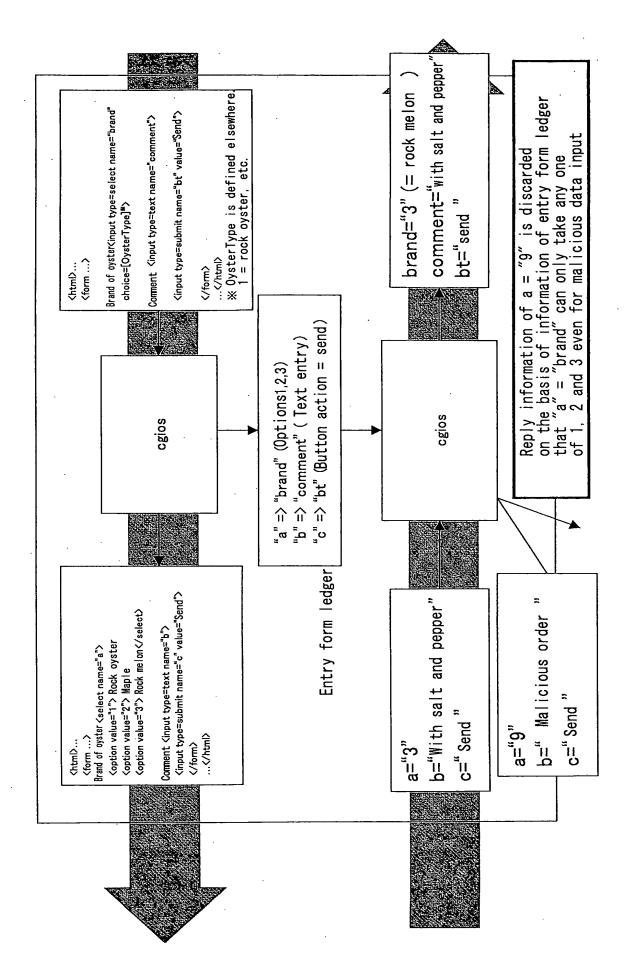


FIG.21

